University of California Santa Cruz
Department of Computer Engineering
Lab Experiment Report # 6
Timing Game

Author: Kyle Jeffrey
Lab Partner: NA
Due Date: 3/3/19

Objective

Using a state machine, implement a top level module that could use two infrared sensors to count the number of times a slug passes through them from left to right or right to left.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18

State Machine:

The state machine served as the brains for the entire game. Giving it several inputs, it routed through different states providing output variables at correct times. The signals TurkeyCountUp, TurkeyCountDown, and TimeCount, told separate counters to count up, down, and start keeping time respectively.

Method:

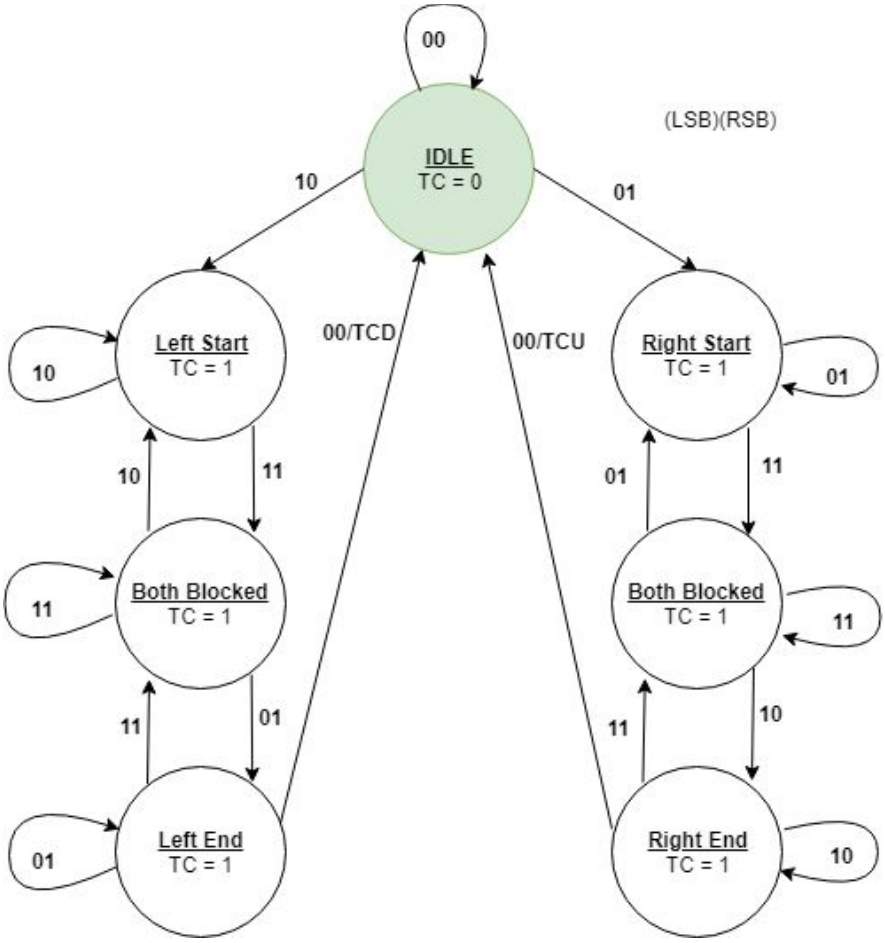
An altered state Machine derivation process was used because of one-hot encoding style flip flop management. There was no need to make a state transition table and derive logic though K-Maps. Starting the same way though, a state diagram was drawn out, and every input arrow was Next State Logic for each state.

States:

- IDLE: The machine sat here until either the left or right sensor was blocked. The state machine is initialized here.
- Right Start: The right sensor is blocked, meaning it's trying to go from right to left. Time Count starts.
- Right Both Blocked: Having started from the right, both sensors are now blocked, Time Count is true.
- Right End: The left sensor is now blocked having started from the right side. If the object then activates no sensors, it means it's passed through to IDLE and the Turkey Count goes Up.
- Left Start: The left sensor is blocked coming from IDLE, meaning the object is trying to go from left to right. Time Count starts.
- Left Both Blocked: Having started from the left, both sensors are now blocked. Time Count is still true.
- Left End: Having started from the left, the right sensor is now blocked, and the object is about to fully pass through to score a point. After this state, if there are then no sensors blocked, Turkey Count goes Down and goes to IDLE state.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18

Results:



INPUTS
- Left Sensor Blocked (LSB)
- Right Sensor Blocked (RSB)

Outputs
- Turkey Count Up (TCU)
- Turkey Count Down (TCD)
- Time Count (TC)

From this, the input and output logic of the states were derived. With one-hot encoding this became very simple:

Next State Logic:

- $D_{IDLE} = \sim LSB \cdot \sim RSB(Q_{IDLE} + Q_{Right\ End} + Q_{Left\ End})$
- $D_{Right\ Start} = \sim LSB \cdot RSB(Q_{IDLE} + Q_{Right\ Start} + Q_{Right\ Both\ Blocked})$
- $D_{Right\ Both\ Blocked} = LSB \cdot RSB(Q_{Right\ Start} + Q_{Right\ Both\ Blocked} + Q_{Right\ End})$
- $D_{Right\ End} = LSB \cdot \sim RSB(Q_{Right\ Both\ Blocked} + Q_{Right\ End})$
- $D_{Left\ Start} = LSB \cdot \sim RSB(Q_{IDLE} + Q_{Left\ Start} + Q_{Left\ Both\ Blocked})$
- $D_{Left\ Both\ Blocked} = LSB \cdot RSB(Q_{Left\ Start} + Q_{Left\ Both\ Blocked} + Q_{Left\ End})$
- $D_{Left\ End} = \sim LSB \cdot RSB(Q_{Left\ Both\ Blocked} + Q_{Left\ End})$

Outputs:

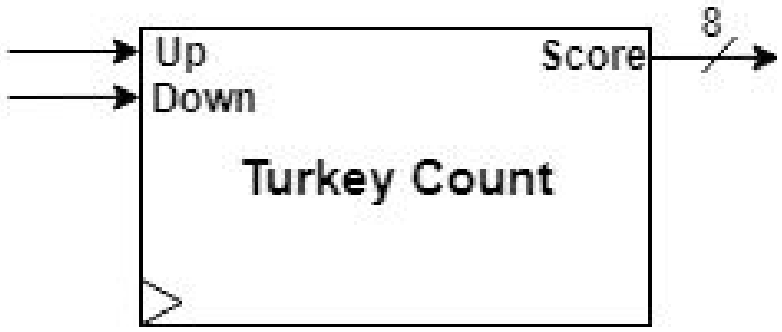
- Time Count = $\sim Q_{IDLE}$
- TurkeyCount Up = $Q_{Right\ End} \cdot \sim LSB \cdot \sim RSB$
- TurkeyCount Down = $Q_{Left\ End} \cdot \sim LSB \cdot \sim RSB$

Turkey Counter:
The turkey counter kept the score of the turkeys passing from right to left and left to right. It receives up and down inputs from the state machine to count. The counter is treated as a two's complement system with 8-bits, the 8th bit being a sign bit.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
5

Method:
The 16-bit counter used from a previous lab was modified to only 8-bits and took in up and down inputs but no Load logic.

RESULTS:



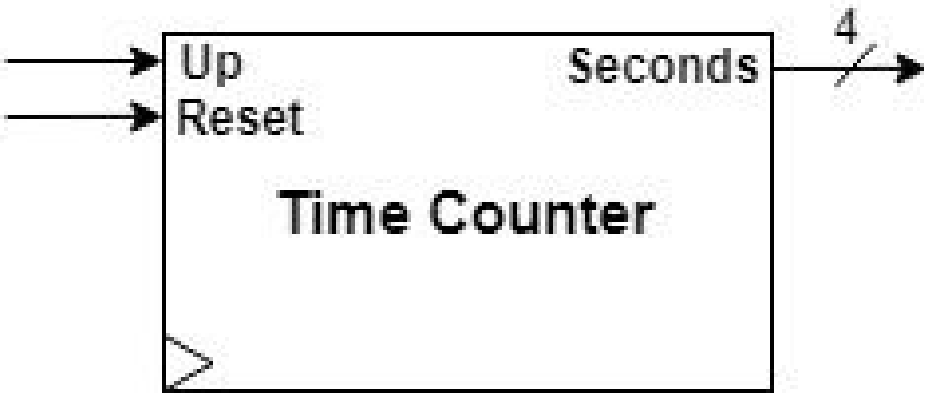
Simulation: The turkey count just needed to operate like a normal counter. Simulating involved providing up and down signals to assure the counter operated correctly.

Time Counter:
The Time Counter only went up to 0xF, and kept track of how many seconds the turkey was in the sensors without fully crossing from one side to the other.

Method:
The design for the counter was grabbed from one of the previous labs that had up, down, and load inputs. The up input was connected to the qsec output from the Lab6_clks module AND'd with TimeCount from the state machine. Load and down were hard coded to zero. The reset for the counter was bound to NOT TimeCount so that any time we didn't want to count, the clock would constantly be resetting. The counter counted every quarter second, so the 3rd bit and up were set to the output to get seconds.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
6

Results:



ERROR: Interesting error within my counter was not setting the down and load control to 0 would break the entire counter.

Two's Complement Converter:

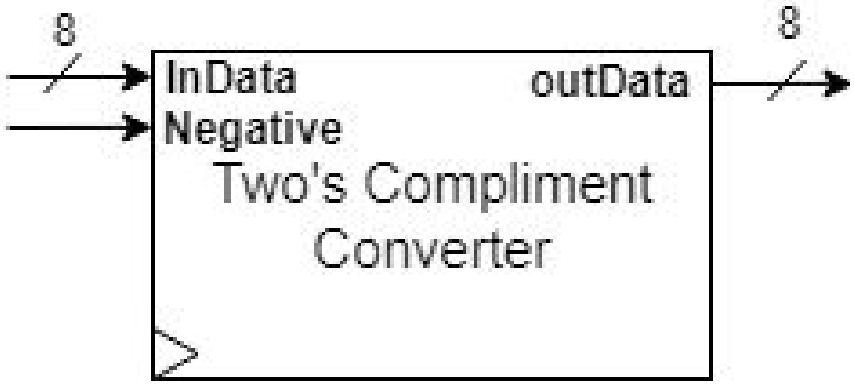
The converter turned the two's complement output from the turkey counter back to a displayable positive version of the negative numbers.

Method:

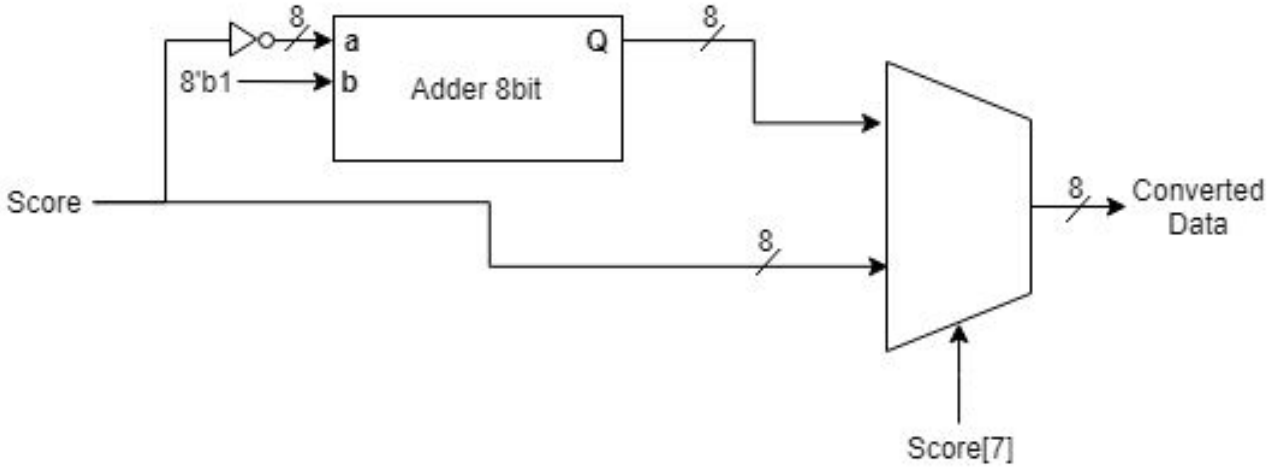
To go from a negative to a positive number and vice versa in two's complement, the number should be bitwise NOT'ed and then add 1. The data was inputted to a custom adder that NOT'd the input and added 1, and the output of that was attached to a 2_1x8 mux that had the unaffected data as the other input. The sign bit would then select which logic to use.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
7

Results:



Two's Compliment Converter



Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
8

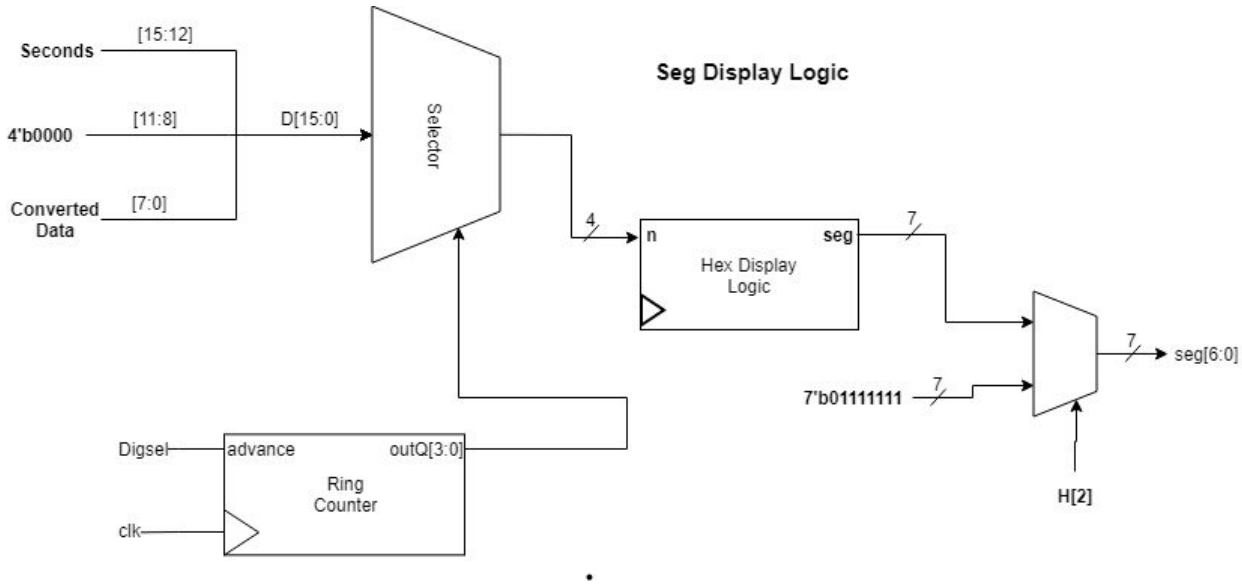
Display Logic:

The display logic was controlled at the top level module and didn't have a discreet module. It controlled turning the Anode's for the hex displays on and off and converting the Time Counter and Turkey Counter to readable hex digits on the 4 Hex displays.

Method:

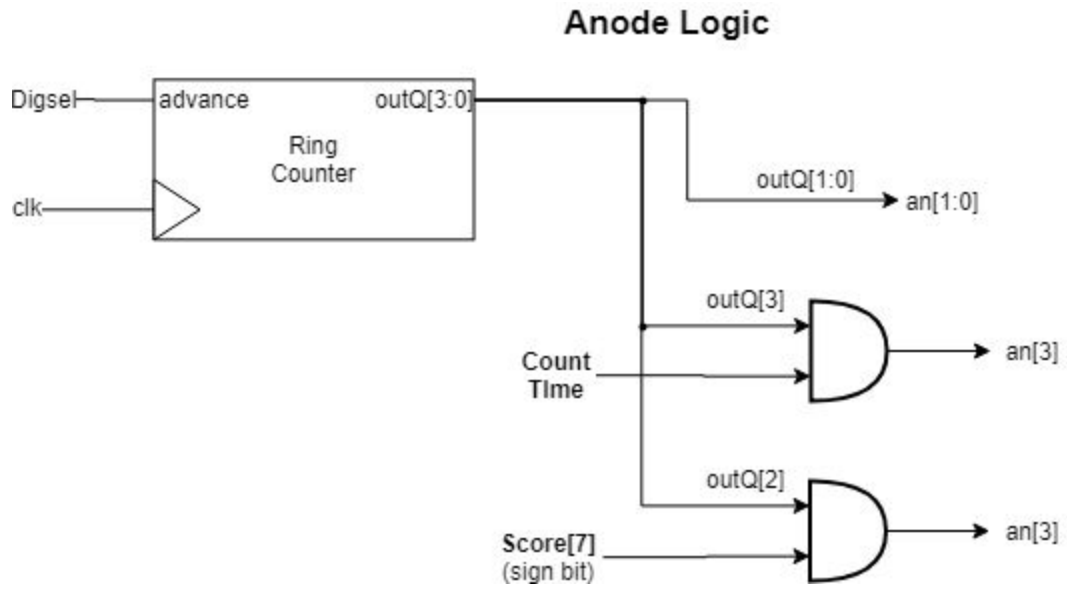
Usually, when displaying to the four hex digits, the ring counter is directly connected to the selector which outputs 4 bits to the hex logic that'll display a 4 bit number. The issue here was that the hex logic contains no logic for displaying a minus sign. To avoid needing to repurpose the hex display logic, a 2_1x8 bit mux was used to either display the minus sign or the regular outputs from the selector. When the ring counter for the third bit was true, the minus sign logic was display, which was the hard coded segment values for displaying a minus sign.

Results:



Seg Display Logic: The segment logic here is a slightly modified version of the normal selector/ring counter display logic, but instead of only using outputs from just the selector, I hard coded the value for the minus sign to be displayed when the ring counter was active low

for the third hex display (H[2]). Ostensibly, all this does is include a minus sign for the display logic for the third hex display, while all other segment values come from the selector.



Active low logic was emitted to make reading easier. MISTAKE: The bottom AND should have output an[2]

Anode Logic: The anode logic here is a slightly modified version of typical anode behavior. The ring counter still controls activating all of the displays, but at certain points we want to hide the displays. The count clock doesn't want to be displayed until the clock actually starts counting which is conveniently provided by the state machine. The negative sign is hardcoded in and doesn't want to be displayed unless the number is negative. This signal is the sign bit from the two's complement version of the score.

Simulation: To test proper function, I just needed to make sure that the minus sign always came through while the ring counter was activating the third hex display. The anode logic just needed to deactivate the third anode if the number was negative and if no sensors were blocked, the fourth anode should be deactivated.

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
10

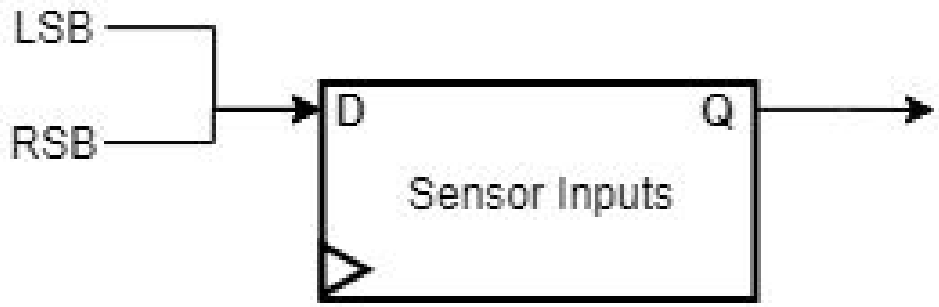
Sensor Inputs:

The pushbuttons were ran through flip flops to synchronize them.

Method:

Using two flipflops, store input the values from the pushbuttons, and then invert the outputs to express the active low value of the logic. Connect these outputs to LED 15 and LED 9.

Results:



Top Level Design:

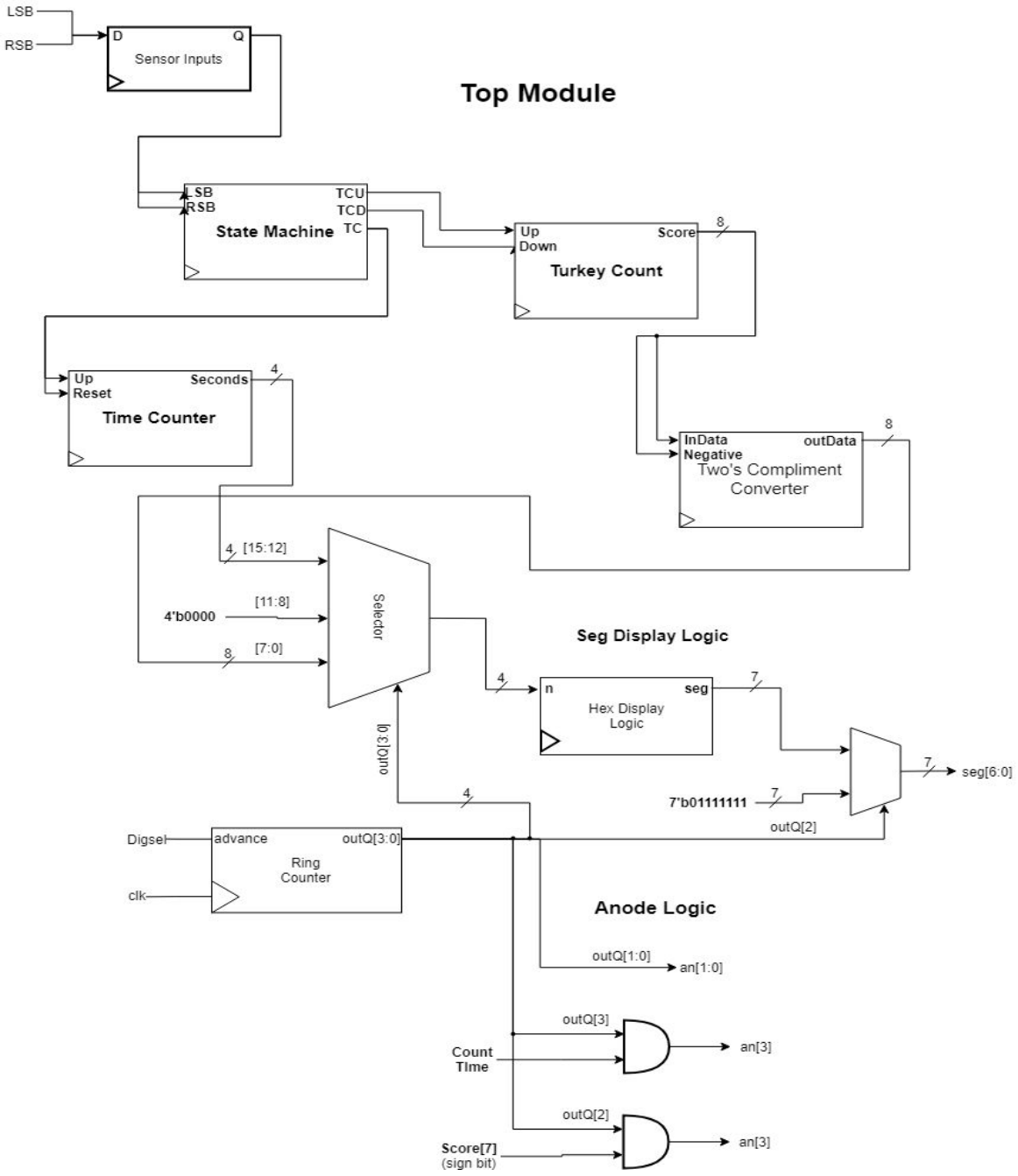
The top level connects everything together. The state machine is the heart of the entire circuit implementation, and almost all of the logic is either feeding the state machine signals or receiving signals from the state machine. The state machine cycles through 7 states providing the outputs for specified amounts of time and values.

Method:

The full assembly began with the state machine and the front end logic. Making sure the correct signals were getting sent to the state machine was most important to me.

Results:

Kyle Jeffrey
Prof Schlag
CE 100
2/21/18

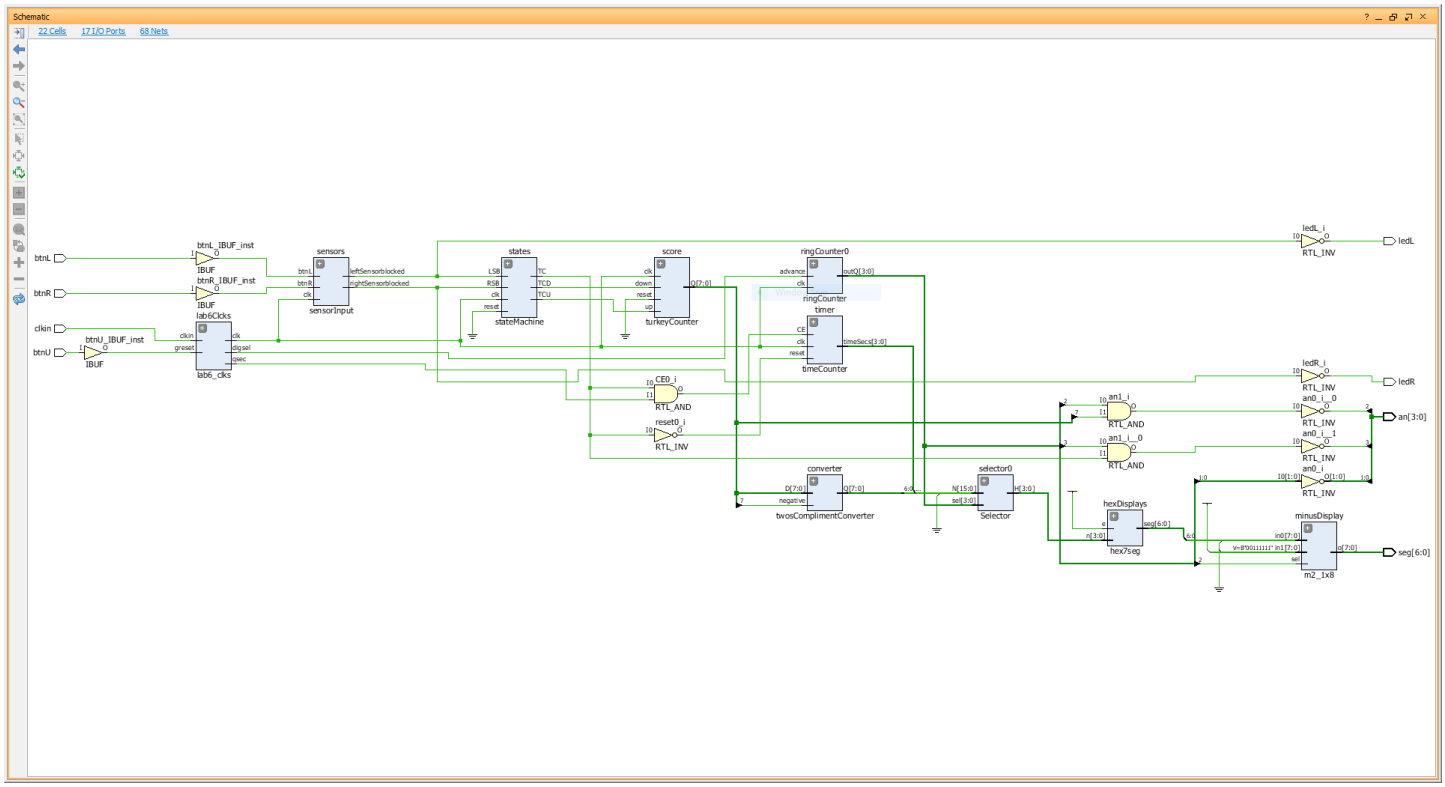


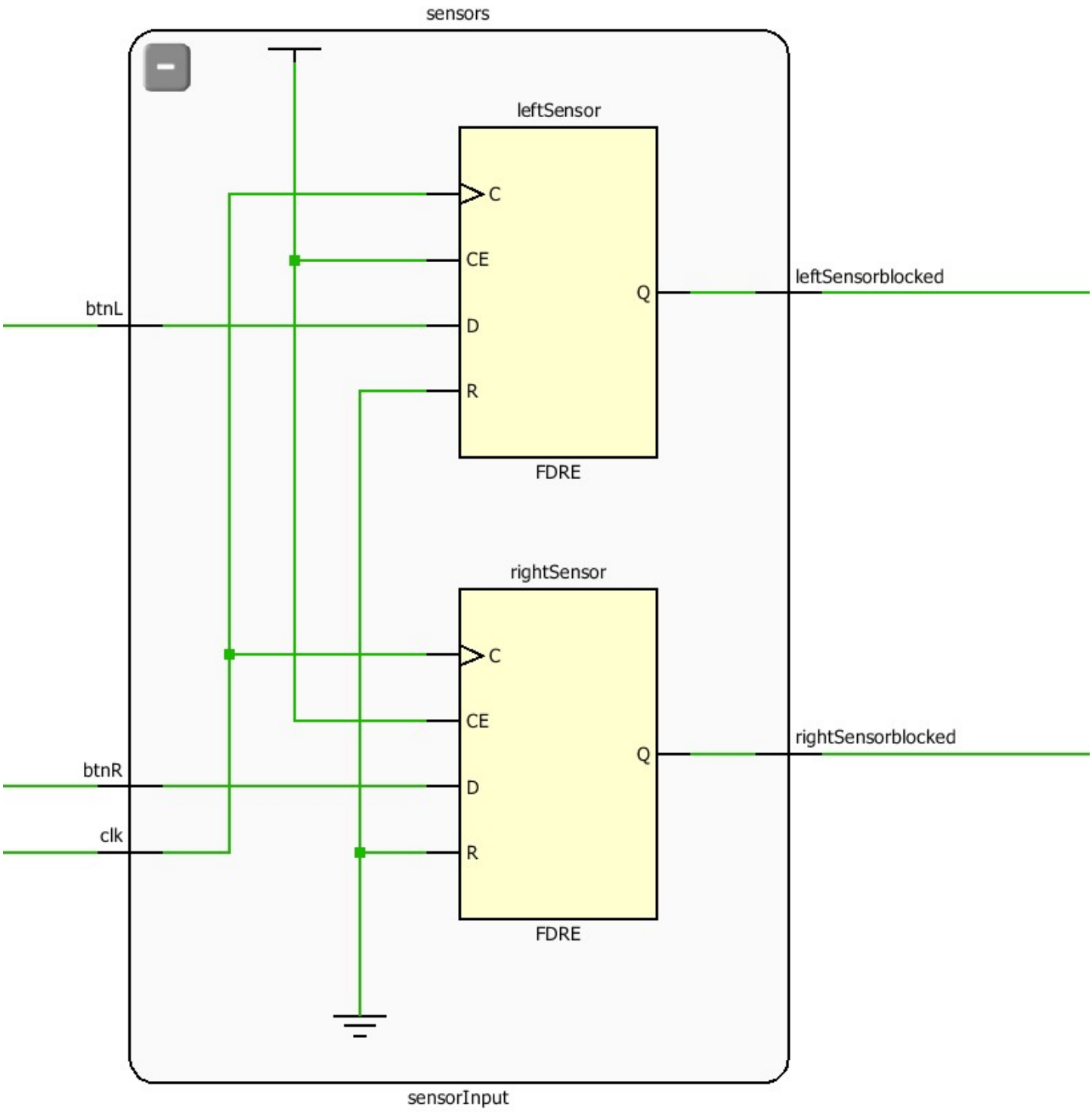
NOTE: The diagram excludes the notion of active low signals for the display logic for ease of schematic flow.

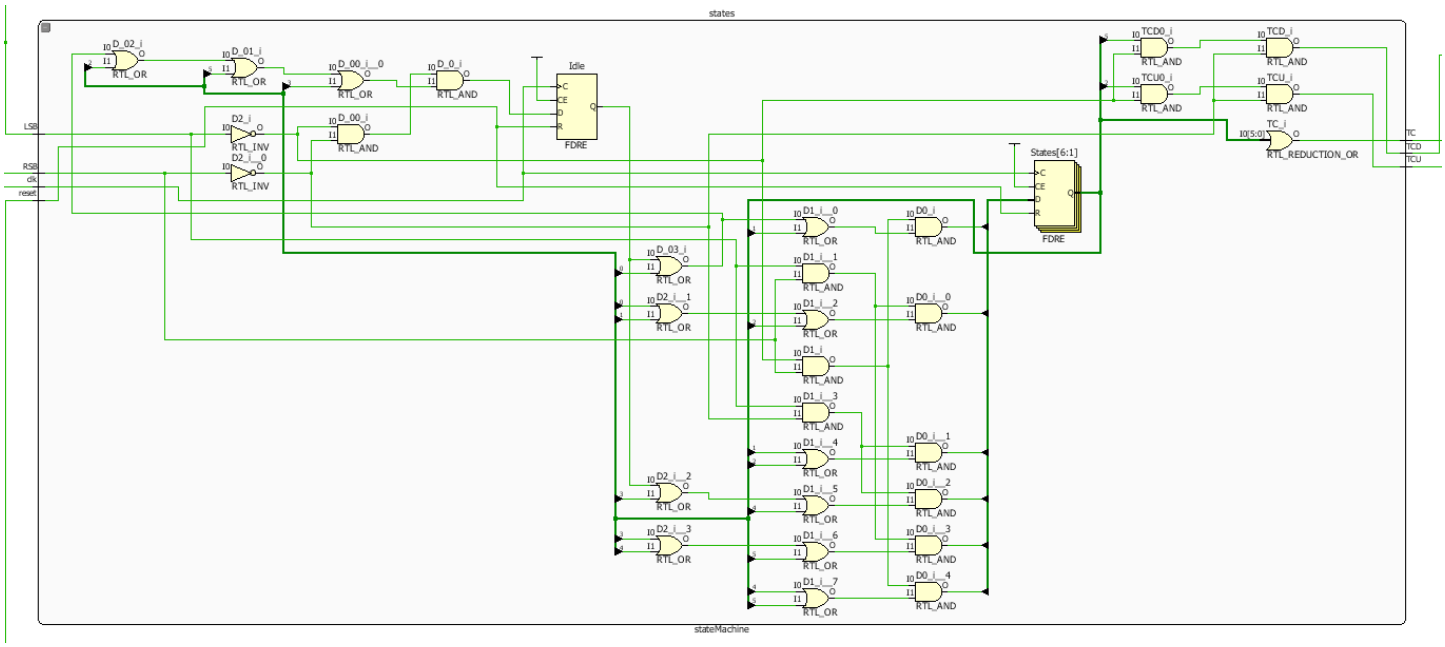
Kyle Jeffrey
Prof Schlag
CE 100
2/21/18
12

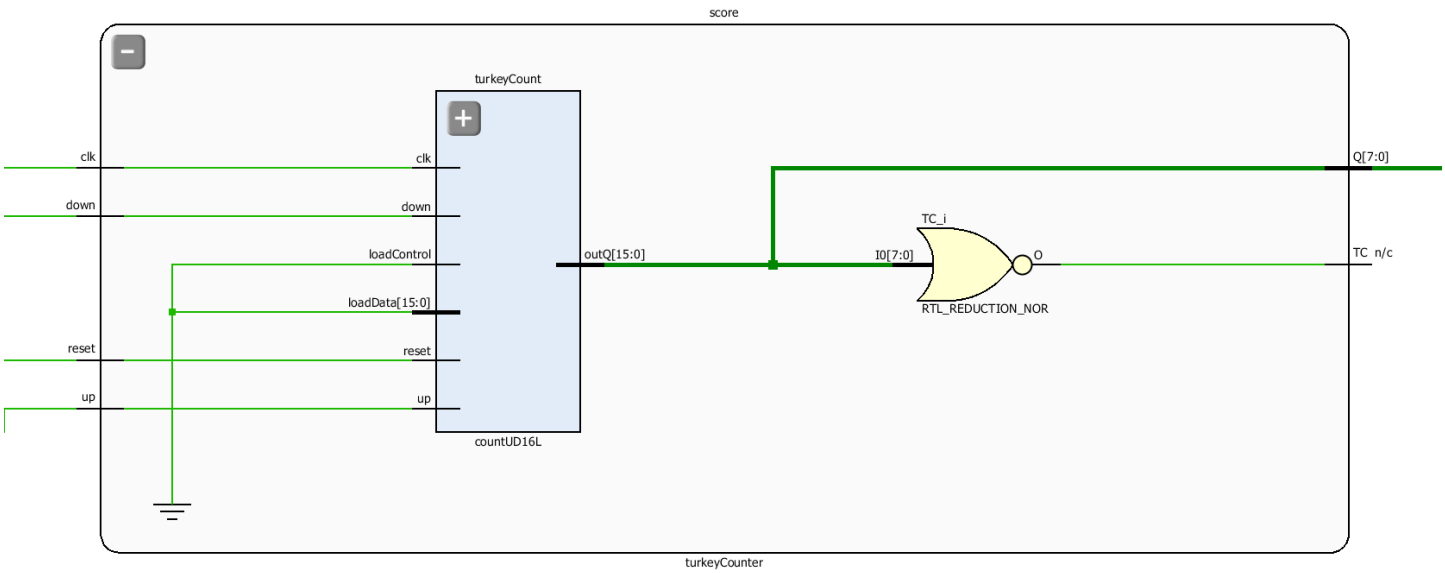
Conclusion

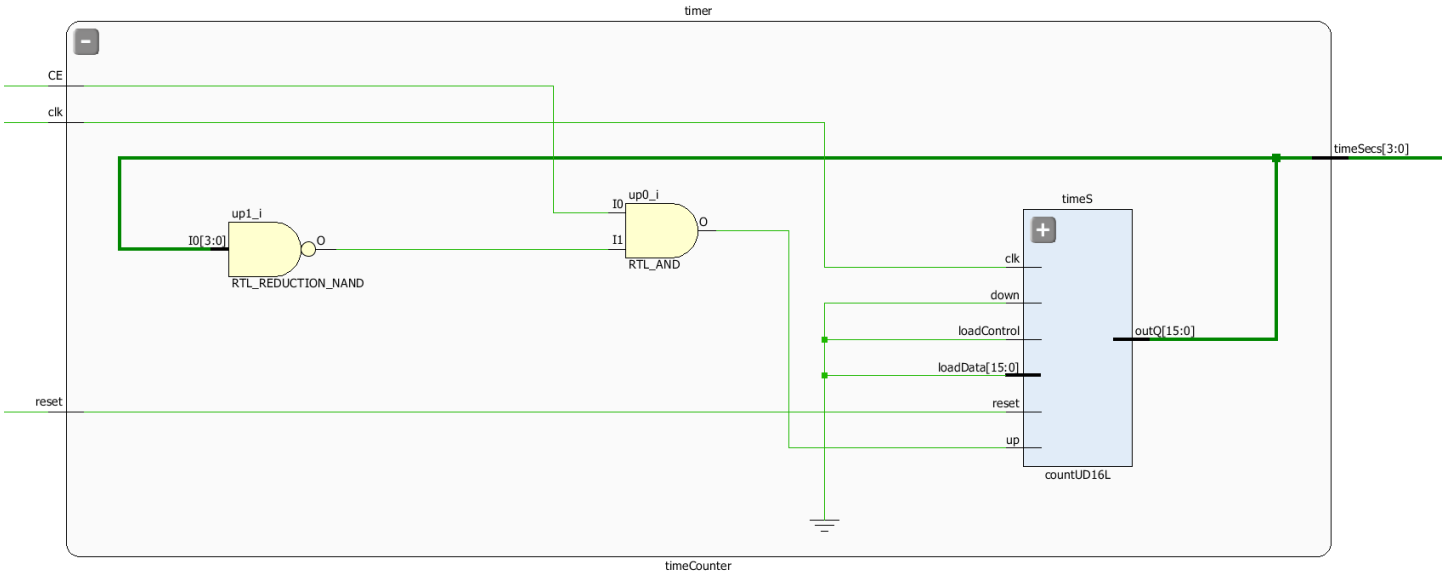
This lab continued the use of state machine logic. I learned new ways to implement the ring counter and hex display logic and how to manually add values in. I definitely could've simplified the logic for the display though. Rather than using a mux and swapping between the two logics for the seg display, I could've adjusted the hex display to print a digit differently if negative. Containing the logic inside of the hex module would've simplified the entire display section of the logic.

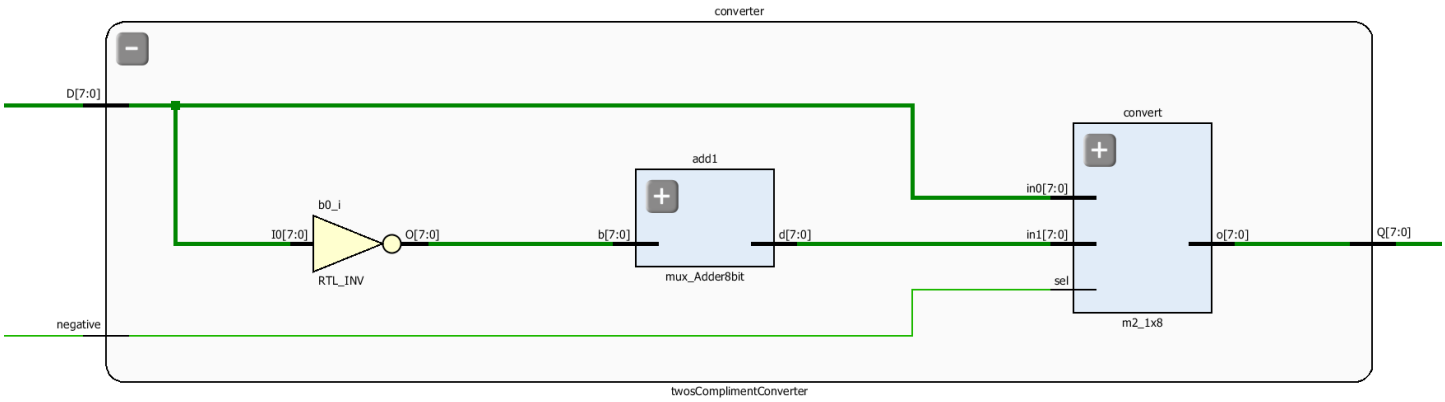

Schematic
22 Cells
17 I/O Ports
68 Nets
btnL
btnR
clkin
btnU
IBUF
btnL_IBUF_inst
btnR_IBUF_inst
btnU_IBUF_inst
lab6Clks
lab6_clks
sensors
sensorInput
leftSensorblocked
rightSensorblocked
states
stateMachine
LSB
RSB
TC
TCD
TCU
score
turkeyCounter
ringCounter0
ringCounter
timer
timeCounter
CE0_i
RTL_AND
reset0_i
RTL_INV
converter
twosComplimentConverter
selector0
Selector
an1_i
an1_i_0
hexDisplays
hex7seg
ledL_i
ledR_i
an0_i_0
an0_i_1
an0_i
minusDisplay
m2_1x8
ledL
ledR
an[3:0]
seg[6:0]


sensors
leftSensor
C
CE
D
R
Q
FDRE
leftSensorblocked
btnL
rightSensor
C
CE
D
R
Q
FDRE
rightSensorblocked
btnR
clk
sensorInput


states
LSB
RSB
clk
reset
D_02_i
RTL_OR
D_01_i
RTL_OR
D_00_i__0
RTL_OR
D_0_i
RTL_AND
Idle
FDRE
D2_i
RTL_INV
D2_i__0
RTL_INV
D_00_i
RTL_AND
D_03_i
RTL_OR
D2_i__1
RTL_OR
D2_i__2
RTL_OR
D2_i__3
RTL_OR
D1_i__0
RTL_OR
D1_i__1
RTL_AND
D1_i__2
RTL_OR
D1_i
RTL_AND
D1_i__3
RTL_AND
D1_i__4
RTL_OR
D1_i__5
RTL_OR
D1_i__6
RTL_OR
D1_i__7
RTL_OR
D0_i
RTL_AND
D0_i__0
RTL_AND
D0_i__1
RTL_AND
D0_i__2
RTL_AND
D0_i__3
RTL_AND
D0_i__4
RTL_AND
States[6:1]
FDRE
TCD0_i
RTL_AND
TCU0_i
RTL_AND
TCD_i
RTL_AND
TCU_i
RTL_AND
TC_i
I0[5:0]
RTL_REDUCTION_OR
TC
TCD
TCU
stateMachine


score
turkeyCount
clk
down
loadControl
loadData[15:0]
reset
up
clk
down
reset
up
countUD16L
outQ[15:0]
TC_i
I0[7:0]
O
RTL_REDUCTION_NOR
Q[7:0]
TC n/c
turkeyCounter


timer
CE
clk
reset
timeSecs[3:0]
up1_i
I0[3:0]
O
RTL_REDUCTION_NAND
up0_i
I0
I1
O
RTL_AND
timeS
clk
down
loadControl
loadData[15:0]
reset
up
outQ[15:0]
countUD16L
timeCounter


converter
D[7:0]
b0_i
I0[7:0]
O[7:0]
RTL_INV
add1
b[7:0]
d[7:0]
mux_Adder8bit
convert
in0[7:0]
in1[7:0]
sel
o[7:0]
m2_1x8
Q[7:0]
negative
twosComplimentConverter

```
`timescale 1ns / 1ps


module turkeyCounter(
    input up, down, clk, reset,

    output [7:0] Q,
    output TC
    );

    wire [15:0] iQ;

    assign Q = iQ[7:0];

    countUD16L turkeyCount(.clk(clk), .up(up), .down(down), .reset(reset), .outQ(iQ),
.loadControl(1'b0));

    assign TC = ~|{Q[7:0]};

endmodule
```

```
`timescale 1ns / 1ps

module stateMachine(
    input LSB, RSB, clk, reset,      //Left Sensor Blocked, Right Sensor Blocked
    output TCU, TCD, TC              // Turkey Count Up, Turkey Count Down, Time Coun
    );


    wire [6:0] D, Q;
    assign D[0] = (~LSB &  ~RSB) & (Q[0]  | Q[1] | Q[3] | Q[6] | Q[4]);

    //Right to left path
    assign D[1] = (~LSB &   RSB) & (Q[0]  | Q[1] | Q[2]);
    assign D[2] = ( LSB &   RSB) & (Q[1]  | Q[2] | Q[3]);
    assign D[3] = ( LSB &  ~RSB) & (Q[2]  | Q[3]);

    //Left to Right path
    assign D[4] = ( LSB &  ~RSB) & (Q[0]  | Q[4] | Q[5]);
    assign D[5] = ( LSB &   RSB) & (Q[4]  | Q[5] | Q[6]);
    assign D[6] = (~LSB &   RSB) & (Q[5]  | Q[6]);




    FDRE #(.INIT(1'b1) ) Idle        (.C(clk), .R(reset), .CE(1'b1), .D(D[0]),
.Q(Q[0]));
    FDRE #(.INIT(1'b0) ) States[6:1] (.C(clk), .R(reset), .CE(1'b1), .D(D[6:1]),
.Q(Q[6:1]));


    //outputs
    assign TCU = Q[3] & ~LSB & ~RSB;
    assign TCD = Q[6] & ~LSB & ~RSB;
    assign TC  = |{Q[6:1]};
endmodule
```

```
`timescale 1ns / 1ps


module sensorInput(
    input clk, btnL, btnR,
    output leftSensorblocked, rightSensorblocked
    );

    FDRE #(.INIT(1'b0) ) leftSensor (.C(clk), .R(reset), .CE(1'b1), .D(btnL),
.Q(leftSensorblocked));
    FDRE #(.INIT(1'b0) ) rightSensor (.C(clk), .R(reset), .CE(1'b1), .D(btnR),
.Q(rightSensorblocked));


endmodule
```

```
`timescale 1ns / 1ps

module topModule(
    input btnL,  btnR, btnU, clkin,
    output [6:0] seg,
    output ledL, ledR,
    output [3:0] an
    );

    wire scoreUp, scoreDown, countTime, scoreZero, clk, qsec, digsel, leftSensor,
rightSensor;
    wire [7:0] scoreQ, displayQ;
    wire [3:0] seconds;

    sensorInput sensors(.clk(clk), .btnL(btnL), .btnR(btnR),
.leftSensorblocked(leftSensor), .rightSensorblocked(rightSensor));

    assign ledL = ~leftSensor;
    assign ledR = ~rightSensor;

    stateMachine states(.LSB(leftSensor), .RSB(rightSensor), .clk(clk), .reset(1'b0),
.TCU(scoreUp), .TCD(scoreDown), .TC(countTime));

    turkeyCounter score(.clk(clk), .up(scoreUp), .down(scoreDown), .reset(1'b0),
.Q(scoreQ), .TC(scoreZero));

    timeCounter timer(.clk(clk), .CE(countTime & qsec), .timeSecs(seconds),
.reset(~countTime));




    twosComplimentConverter converter(.D(scoreQ), .Q(displayQ),
.negative(scoreQ[7])); // bit 8 of score is sign bit

    lab6_clks lab6Clcks(.clkin(clkin), .clk(clk), .greset(btnU), .digsel(digsel),
.qsec(qsec));



    //DISPLAY LOGIC
    wire [3:0] H, selQ;
    wire [6:0] p, negSign;

    assign negSign = ~{1'b1,1'b0,1'b0,1'b0,1'b0,1'b0,1'b0};
    ringCounter ringCounter0(.clk(clk), .advance(digsel), .outQ(H));
```

```
    Selector selector0(.N({seconds, 4'b0000 , {1'b0, displayQ[6:0]} }), .sel(H),
.H(selQ));

    hex7seg hexDisplays(.n(selQ), .e(1'b1), .seg(p));

    m2_1x8 minusDisplay(.in0(p) ,.in1(negSign), .sel(H[2]), .o(seg));

    assign an[3]   = ~ (H[3] & countTime);
    assign an[2]   = ~(H[2]& scoreQ[7]) ;
    assign an[1:0] = ~ H[1:0];


endmodule
```

```
`timescale 1ns / 1ps


module twosComplimentConverter(
    input [7:0] D,
    input negative,
    output [7:0] Q
    );

    wire [7:0] negQ;

    //BITWISE NOT AND ADD 1
    mux_Adder8bit add1(.b(~D), .d(negQ));

    m2_1x8 convert(.in0(D), .in1(negQ), .sel(negative), .o(Q));

endmodule
```

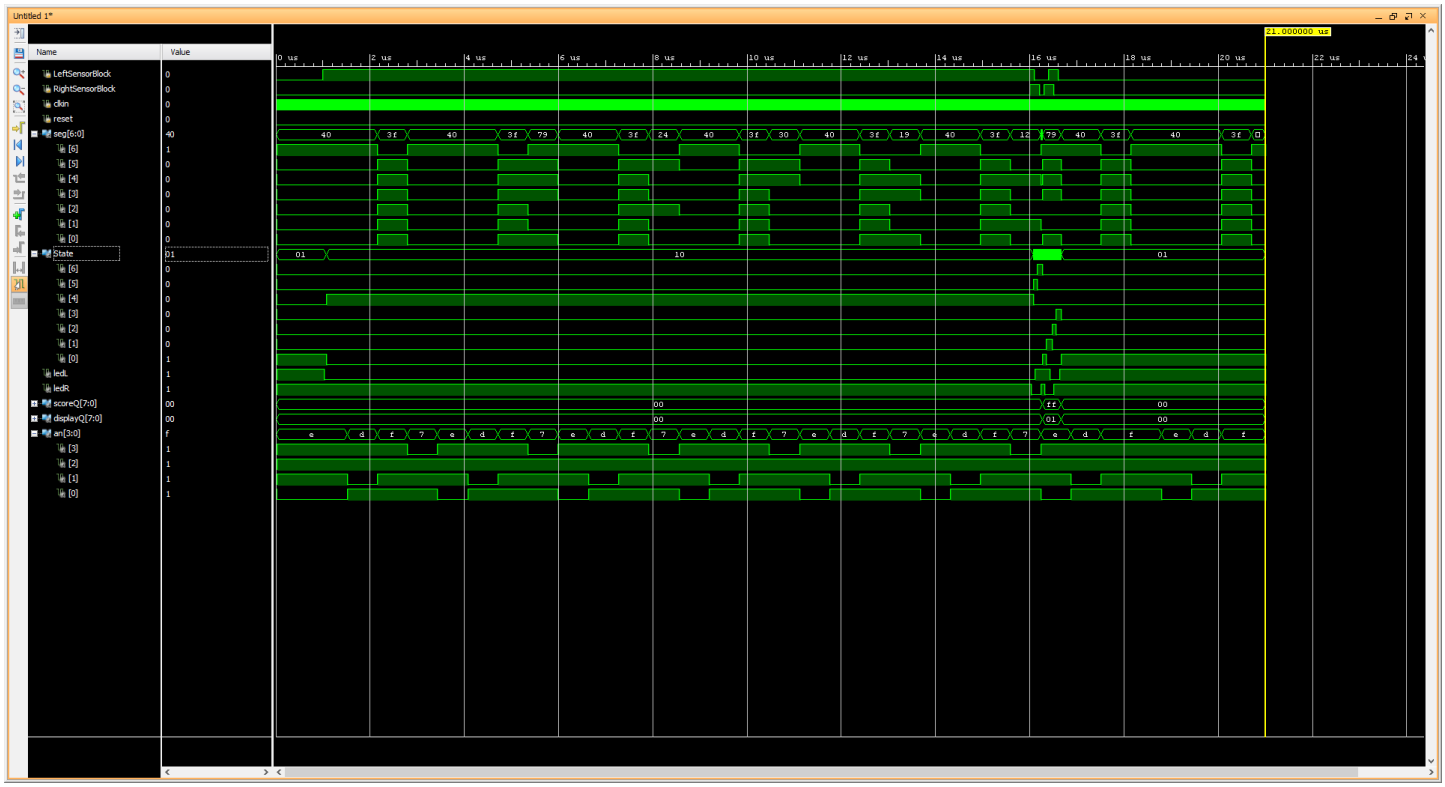
```
`timescale 1ns / 1ps


module timeCounter(
    input clk, CE, reset,         // When Clock Enable is high, timer starts counting
    output [3:0] timeSecs                  //Time
    );

    wire [15:0] timeQsecs;
    countUD16L timeS  (.clk(clk), .reset(reset), .up(CE & ~&{timeQsecs[5:2]}),
.outQ(timeQsecs), .loadControl(1'b0), .down(1'b0));

    assign timeSecs = timeQsecs[5:2];

endmodule
```

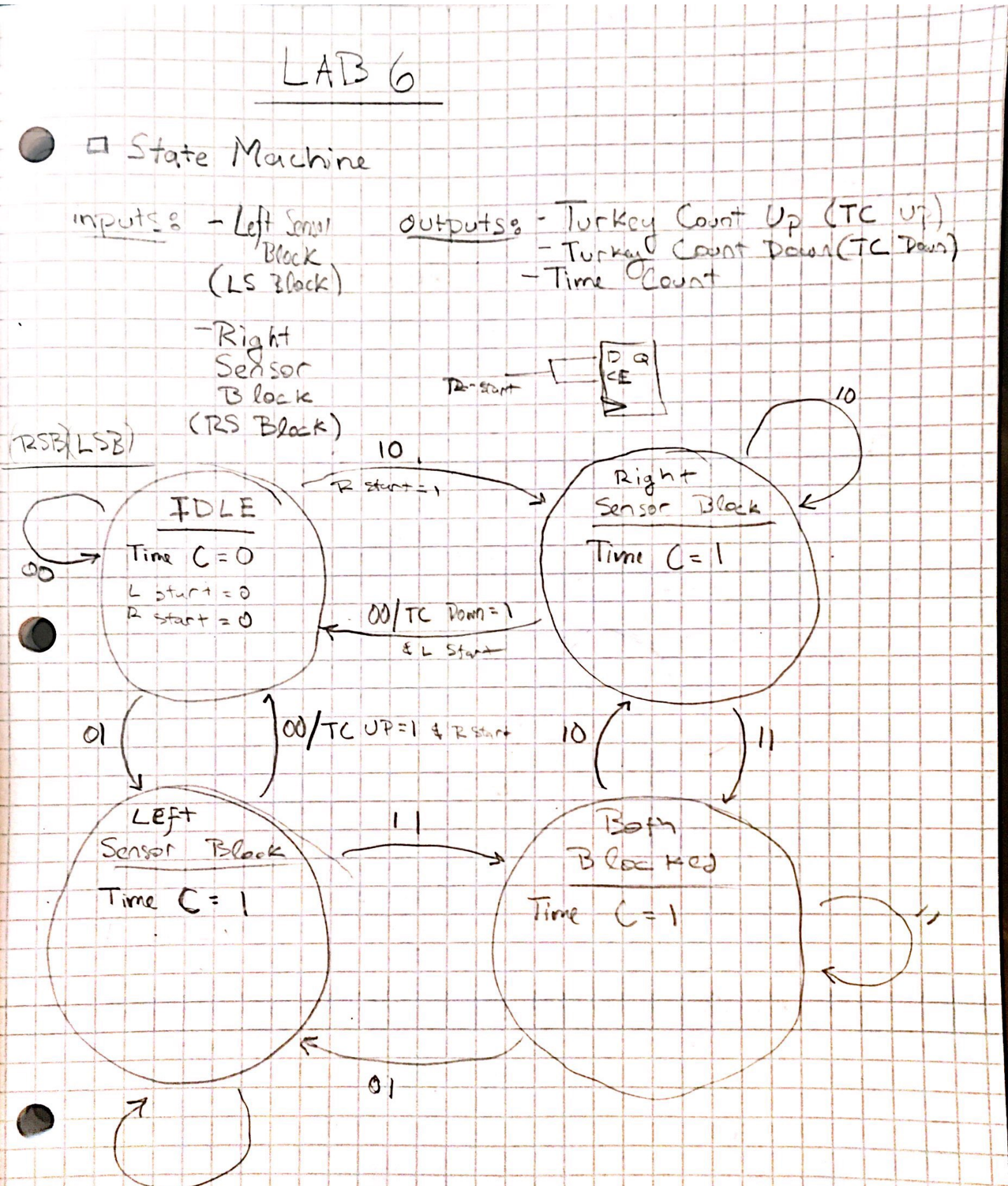

Untitled 1*
Name
Value
LeftSensorBlock
RightSensorBlock
clkin
reset
seg[6:0]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
State
[6]
[5]
[4]
[3]
[2]
[1]
[0]
ledL
ledR
scoreQ[7:0]
displayQ[7:0]
an[3:0]
[3]
[2]
[1]
[0]
0 us
2 us
4 us
6 us
8 us
10 us
12 us
14 us
16 us
18 us
20 us
22 us
24 us
21.000000 us

LAB 6

State Machine

inputs: - Left Sensor Block (LS Block)
- Right Sensor Block (RS Block)

outputs: - Turkey Count Up (TC Up)
- Turkey Count Down (TC Down)
- Time Count



$Q_3 = Q_{IDLE}$

$Q_2 = Q_{RSB\ block}$

$Q_1 = Q_{Both\ block}$

$Q_0 = Q_{LSB\ block}$

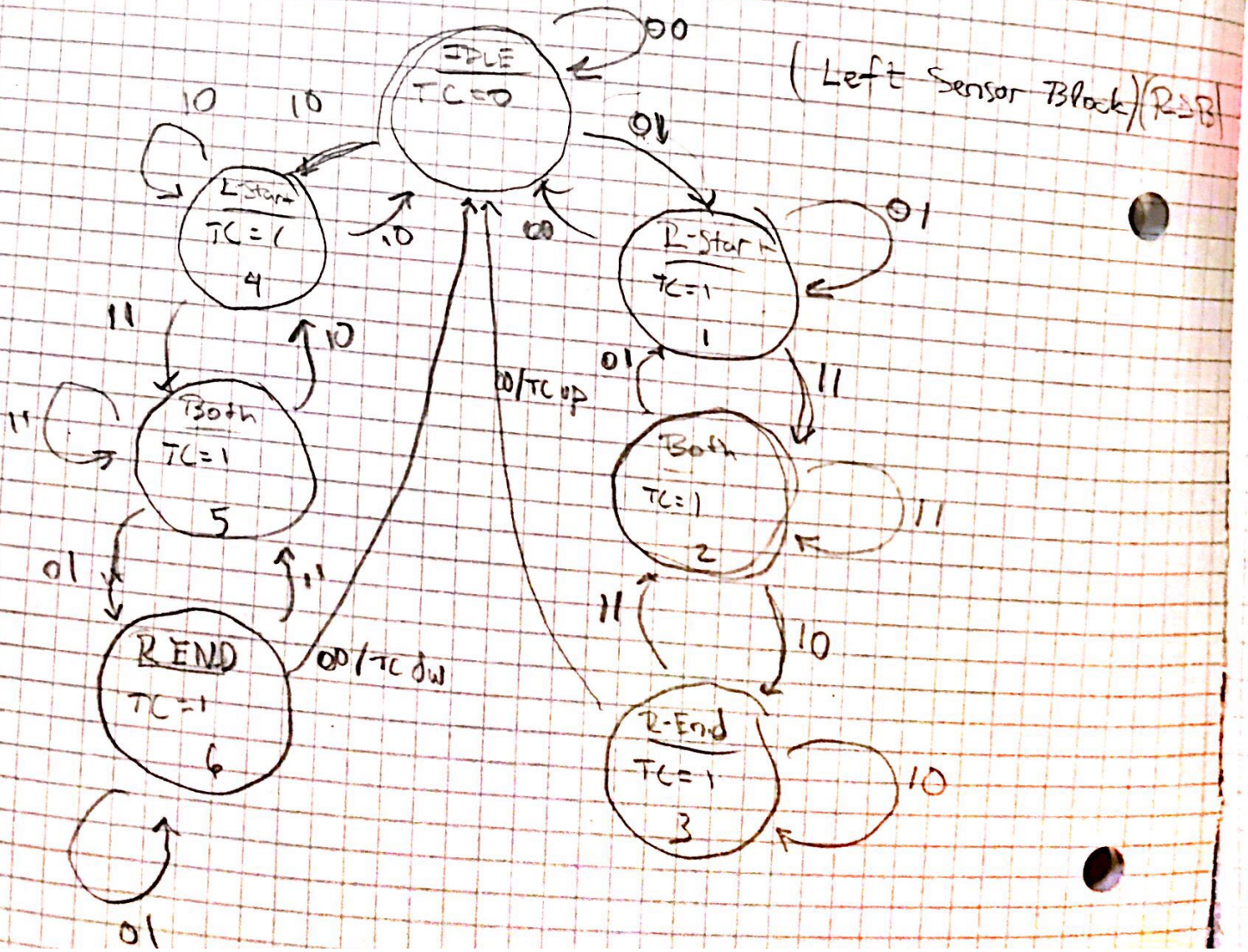
$D_{IDLE} = Q_{LSB} \cdot \overline{LSB} \cdot \overline{RSB} + Q_{RSB} \cdot \overline{LSB} \cdot \overline{RSB} + \overline{LSB} \cdot \overline{RSB}\, Q_{IDLE}$

$D_{RSB} = Q_{START} \cdot \overline{LSB} \cdot RSB + Q_{BB} \cdot \overline{LSB} \cdot RSB + Q_{RSB} \cdot \overline{LSB} \cdot RSB$

$D_{Both} = Q_{RSB} \cdot LSB \cdot RSB + Q_{LSB} \cdot LSB \cdot \overline{RSB} + Q_{BB} \cdot LSB \cdot RSB$

$D_{LSB} = Q_{IDLE} \cdot LSB \cdot \overline{RSB} + Q_{BB} \cdot LSB \cdot \overline{RSB} + Q_{LSB} \cdot LSB \cdot \overline{RSB}$

(Left Sensor Block)(RSB)



Q_0 = IDLE

$Q[1:3]$ = R-Start, R Both, R-End

$Q[4:6]$ = L-Start, L Both, L-End

$D_0 = Q_0 \cdot \bar{L} \cdot \bar{R} + Q_3 \cdot \bar{L} \cdot \bar{R} + Q_6 \cdot \bar{L} \cdot \bar{R} + Q[4]$

(R-Start) $D_1 = Q_0 \cdot \bar{L} \cdot R + Q_1 \cdot \bar{L} \cdot R + Q[2]$

(R Both) $D_2 = Q_1 \cdot L \cdot R + Q_2 \cdot L \cdot R + Q_3$

(R-End) $D_3 = Q_2 \cdot L \cdot \bar{R} + Q_3 \cdot L \cdot \bar{R}$

(L-Start) $D_4 = Q_0 \cdot L\bar{R} + Q_4 L\bar{R} + Q_5$

(L-Both) $D_5 = Q_4 LR + Q_5 LR + Q_6$

(L-End) $D_6 = Q_5 \bar{L}R + Q_6 \bar{L}R$

TC up = $Q_3 \cdot \bar{L} \cdot \bar{R}$ TC = $Q[6:1]$

TC dwn = $Q_6 \cdot \bar{L} \cdot \bar{R}$

Time Count

- Use up counter with 1 second clock

quarter sec 0000
second

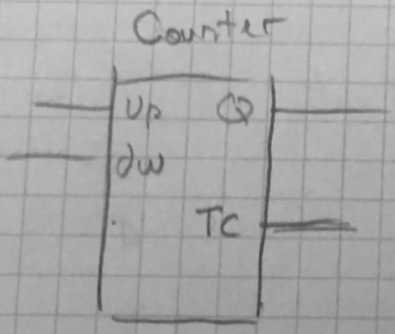
Turkey Counter

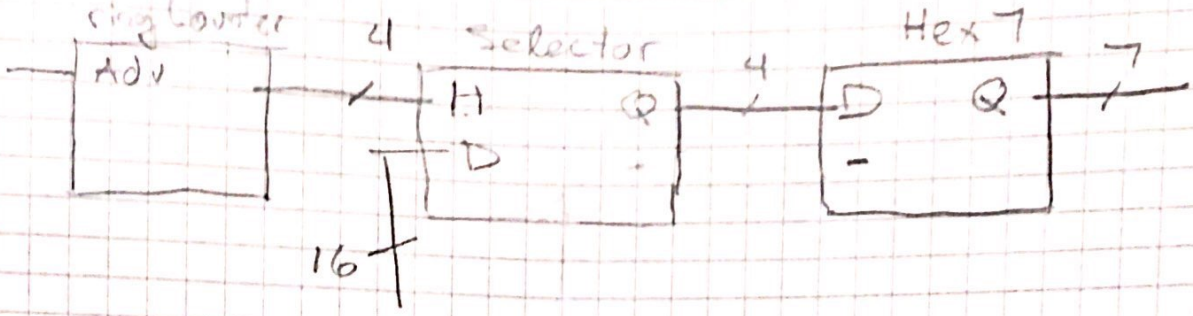
Displaying negative : (FFFF) FLIP, ADD 1

NEED < 0 display logic

8 bits → 3F

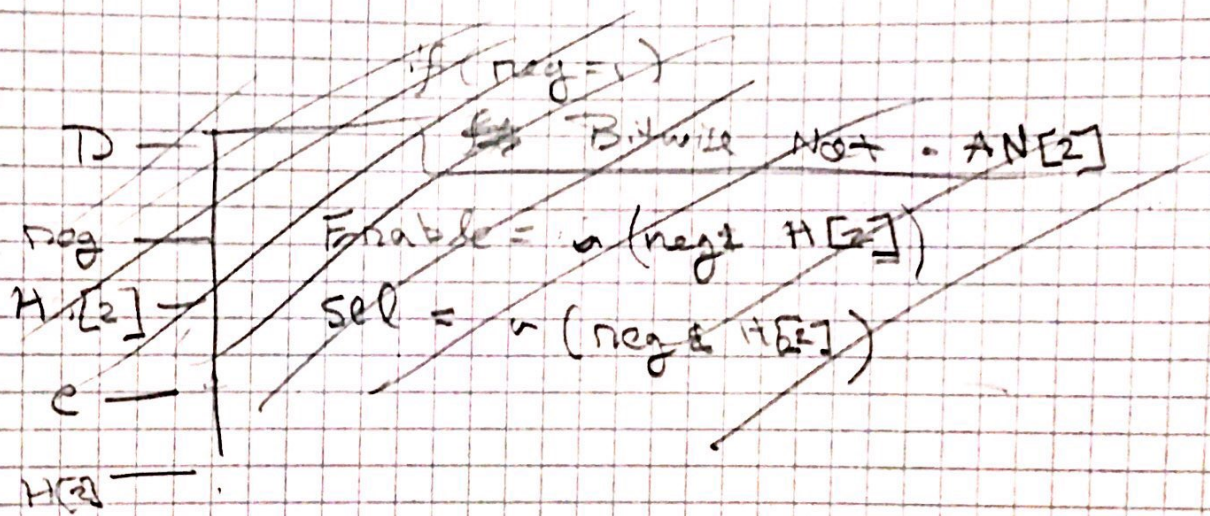
sign bit





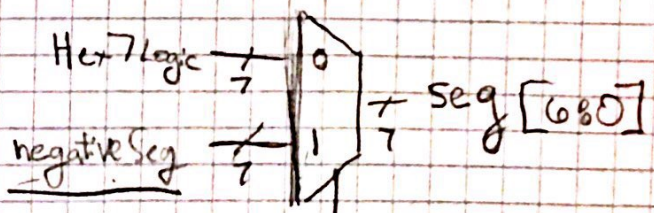
seg[6:0]

modify



Ring Counter

H: 1000 } digsel
0100
0010
0001



H[2] = an[2]

an[2] = H[2] & scoreQ[7]